Evaluating the Impact of High-Bandwidth Memory on MPI Communications

Giuseppe Congiu[*], Pavan Balaji
Mathematics and Computer Science Division
Argonne National Laboratory
email: gcongiu@anl.gov, balaji@anl.gov

I. Introduction

High-end computing clusters are becoming increasingly heterogeneous and include a variety of compute components, ranging from standard CPUs to highly parallel accelerators. Such increasing heterogeneity is driven by the need to better exploit data-level parallelism in existing and future codes and to account for a paradigm shift in high-performance computing (HPC) workloads that are moving from simulations to more complex workflows involving data ingestion, generation, and analysis, for example, deep learning applications [1].

The massive parallelism offered by accelerators exacerbates the performance gap between computation and memory, with the memory subsystem that has to supply (absorb) data into (from) an ever-larger number of cores concurrently. For this reason the memory architecture of HPC systems is transitioning from homogeneous configurations, characterized by single DRAM technology, to heterogeneous configurations comprising standard DRAM as well as on-package high-bandwidth memory (HBM) based on 3D stacked DRAM technology [2]. Future architectures will likely include a range of diverse memory components such as HBMs and byte-addressable nonvolatile memories.

Accelerators that integrate HBMs include the Intel Knights Landing (KNL) processors [3], the second generation of the Xeon Phi Many Integrated Core architecture, and the NVIDIA Tesla P100 [4]. Memory heterogeneity exposed by these accelerators introduces further complexity that has to be taken into account by programmers when striving for high performance in their codes. In fact, HBMs are limited in capacity compared with DRAM memory, and not all applications can equally benefit from them; overall, performance strongly depends on the way data is laid out and accessed.

Previous studies [5] [6] [7] have shown that the performance of bandwidth-bound applications can be improved whenever the HBM's capacity is sufficient to accommodate sensitive data structures. If memory requirements cannot be fully satisfied, the user has to decide what data placement strategy to adopt in order to achieve the best results. All these studies, however, lack a fine-grained analysis of the effects that HBMs have on communication libraries such as MPI. Indeed, MPI internally allocates and manages memory objects for a variety of reasons, including efficient intranode communication support. In this case the way the library places its internal objects in memory becomes relevant from a performance standpoint. If on the one hand appropriate placement of MPI objects in HBM can boost the library performance, on the other hand this carves into the available memory budget, reducing the amount of space accessible to the user's data and potentially degrading the overall performance.

To fill the gap left in the literature, we have studied the impact that HBMs have on MPI communications and corresponding memory usage. Our contributions to the state of the art are as follows.

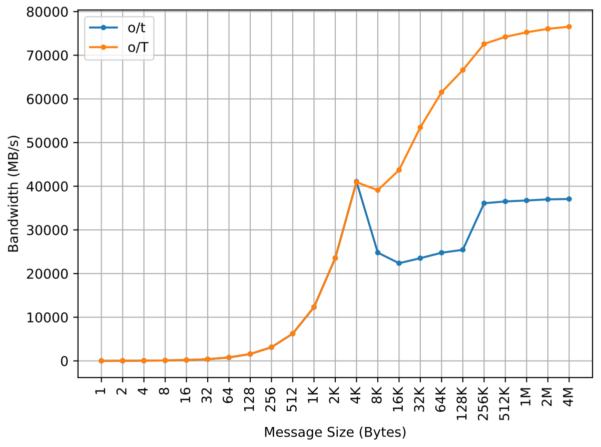
- We have prototyped HBM support into MPI for different types of intranode communication mechanisms, including point-to-point (pt2pt) and remote memory access (RMA). We have used a combination of microbenchmarks and miniapplications to evaluate the effect of HBM on performance for different memory placement strategies of MPI internal objects.
- Based on our findings, we have compiled a list of recommendations for programmers that will be useful for further tuning their codes through optimal configuration of the MPI library. Our recommendations depend on both the analysis of performance-critical memory objects, along with their placement in physical memory, and the resulting memory usage requirements. This last aspect is particularly important because it can also affect the application's overall performance.

In our study we have used the Intel KNL processor, available in the Joint Laboratory for System Evaluation [8] at Argonne National Laboratory, and the MPICH [9] implementation of the MPI standard, developed at Argonne.

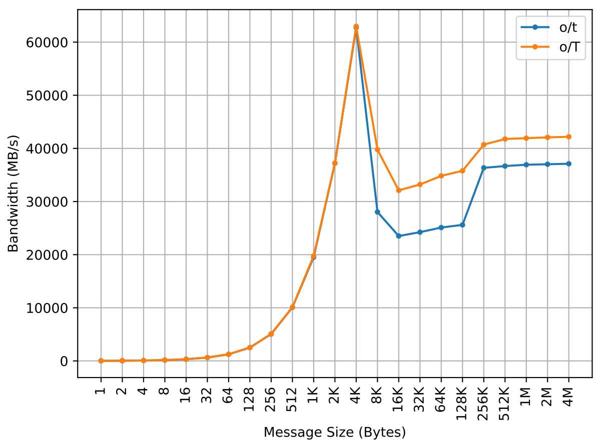
II. Evaluation

The CH3 device in MPICH supports shared-memory communication for both pt2pt and RMA operations. For pt2pt the library uses two separate protocols: *Eager* for short messages and *Rendezvous* for long ones. Eager employs pre-allocated shared-memory buffers called *Fastboxes* and non-blocking message queues (or *Cells*), while rendezvous also employs apposite buffers (*Copy Buffers*) created for the operation and destroyed afterwards. RMA operations expose a separate memory region (or window) that can be allocated in shared memory when `MPI_Win_allocate` or `MPI_Win_allocate_shared` is used.

We evaluated the impact of HBM on intranode shared-memory communication by allocating the aforementioned buffer resources in KNL Multi-Channel DRAM (MCDRAM).



(a)



(b)

Fig. 1: Benchmark results for osu_put_mbw (1a) and osu_get_mbw (1b) for 32 pairs. Lowercase indicates DRAM placement and uppercase MCDRAM placement.

Figure 1 shows bandwidth performance for OSU microbenchmarks multi-`MPI_Put` (1a) and `MPI_Get` (1b) tests when 32 pairs of processes are used. The two tests are similar: put operations load data from the user buffer (*origin*) and store it in the shared-memory window buffer (*target*), and get operations load data from the shared-memory window buffer and store it to the user buffer. However, the most performance-critical object for RMA operations is the store buffer (i.e., the target buffer in put operations and the origin buffer in get operations). In fact, placing this in MCDRAM can boost the bandwidth of put operations by up to 105%. Similar performance improvements can be measured for multilatency tests (not shown).

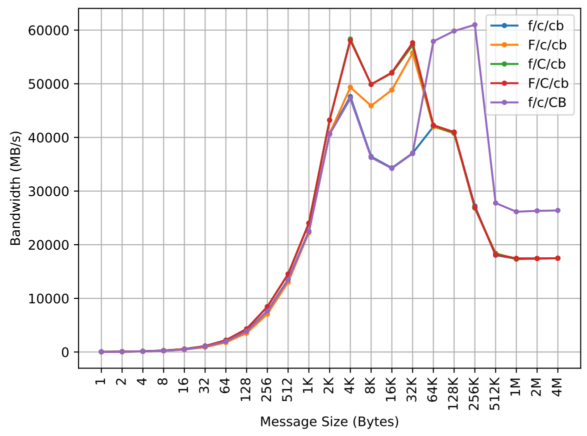


Fig. 2: Benchmark results for osu_mbw_mr for 32 pairs. Here “f” stands for fastboxes, “c” for cells, and “cb” for copy buffers. Lowercase indicates DRAM placement and uppercase MCDRAM placement.

Pt2pt operations have more limited performance improvements because data has to be first copied into the preallocated shared-memory buffers inside the library and from there to the target user buffers (Figure 2). Again, similar performance can be measured for multilatency tests (not shown).

According to the microbenchmarks results and taking into consideration the limited capacity of MCDRAM (only 16 GB in this case), the following recommendations can be made. For RMA, memory should be allocated by using MPI allocation functions, as described above, whenever possible since communication can be greatly improved (especially for put operations). For pt2pt communication, memory cannot be allocated and freed by the program itself but is instead preallocated by the library at init time. In this case, the library memory footprint has to be understood and taken into consideration beforehand.

References

[1] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew, “Deep learning with COTS HPC systems,” in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 June 2013, pp. 1337–1345. [Online]. Available: http://proceedings.mlr.press/v28/coates13.html

[2] G. H. Loh, “3D-stacked memory architectures for multi-core processors,” in *2008 International Symposium on Computer Architecture*, June 2008, pp. 453–464.

[3] A. Sodani, R. Gramunt, J. Corbal, H. S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y. C. Liu, “Knights Landing: Second-generation Intel Xeon Phi product,” *IEEE Micro*, vol. 36, no. 2, pp. 34–46, March 2016.

[4] NVIDIA, “NVIDIA Tesla P100,” https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf, 2016.

[5] L. Oden and P. Balaji, “Hexe: A toolkit for heterogeneous memory management,” in *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, Dec. 2017, pp. 656–663.

[6] I. B. Peng, S. Markidis, E. Laure, G. Kestor, and R. Gioiosa, “Exploring application performance on emerging hybrid-memory supercomputers,” in *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Dec. 2016, pp. 473–480.

[7] C. DeTar, D. Doerfler, S. Gottlieb, A. Jha, D. Kalamkar, R. Li, and D. Toussaint, “MILC staggered conjugate gradient performance on Intel KNL,” p. 270, 12 2016.

[8] “Joint Laboratory for System Evaluation, JLSE cluster,” http://www.jlse.anl.gov/.

[9] “MPICH: A high performance and widely portable implementation of the Message Passing Interface (MPI) standard,” www.mpich.org.